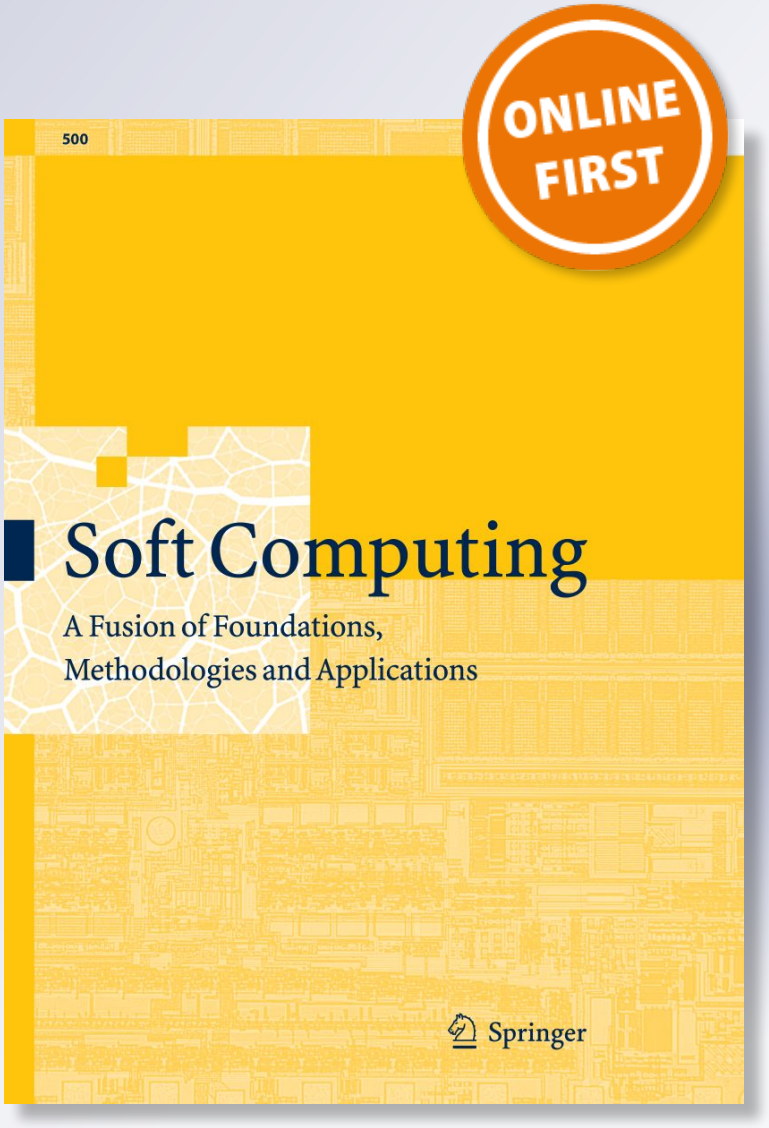
A novel task scheduling scheme in a cloud computing environment using hybrid biogeography-based optimization

Zhao Tong, Hongjian Chen, Xiaomei Deng, Kenli Li & Keqin Li

Soft Computing
A Fusion of Foundations, Methodologies and Applications

ISSN 1432-7643

Soft Comput
DOI 10.1007/s00500-018-3657-0



Springer

Soft Computing
https://doi.org/10.1007/s00500-018-3657-0

METHODOLOGIES AND APPLICATION

A novel task scheduling scheme in a cloud computing environment using hybrid biogeography-based optimization

Zhao Tong[1] · Hongjian Chen[1] · Xiaomei Deng[1] · Kenli Li[2,3] · Keqin Li[2,3,4]

Abstract
Task scheduling, which plays a crucial role in cloud computing and is the critical factor influencing the performance of cloud computing, is an NP-hard problem that can be solved with a heuristic algorithm. In this paper, we propose a novel heuristic algorithm, called biogeography-based optimization (BBO), and a new hybrid migrating BBO (HMBBO) algorithm, which integrates the migration strategy with particle swarm optimization (PSO). Both methods are proposed to solve the problem of scheduling-directed acyclic graph tasks in a cloud computing environment. The basic idea of our approach is to exploit the advantages of the PSO and BBO algorithms while avoiding their drawbacks. In HMBBO, the flight strategy under the BBO migration structure is hybridized to accelerate the search speed, and HEFT_D is used to evaluate the task sequence. Based on the WorkflowSim, a comparative experiment is conducted with the makespan of task scheduling as the objective function. In HMBBO, the flight strategy under the BBO migration structure is hybridized to accelerate the search speed, and HEFT_D is used to evaluate the task sequence. Based on the WorkflowSim, a comparative experiment is conducted with the makespan of task scheduling as the objective function. Both simulation and real-life experiments are conducted to verify the effectiveness of HMBBO. The experiment shows that compared with several classic heuristic algorithms, HMBBO has advantages in terms of global search ability, fast convergence rate and a high-quality solution, and it provides a new method for task scheduling in cloud computing.

Keywords Biogeography-based optimization · Cloud computing · Directed acyclic graph · Task scheduling · WorkflowSim

1 Introduction

Cloud computing is a resource- and service-managing IT commercial model that is derived from parallel computing and grid computing (Mei et al. 2015). In the age of big data, cloud computing platforms provide services to process big data and implement elastic computing. Task scheduling is an NP-hard problem (Lo 1988) that plays a critical role in traditional parallel computing and grid computing. It also exists in the dynamic cloud computing environment and is even more complicated. As the data volume from all fields increases sharply, the task size is also rapidly increasing. There is no particularly useful method for the special nature of the task scheduling problem. Guided-random-search-based (also called meta-

Communicated by V. Loia.

✉ Zhao Tong
tongzhao@hunnu.edu.cn

Hongjian Chen
cchloechj@hotmail.com

Xiaomei Deng
dengxm@smail.hunnu.edu.cn

Kenli Li
lkl@hnu.edu.cn

Keqin Li
lik@newpaltz.edu

[1] College of Information Science and Engineering, Hunan Normal University, Changsha, Hunan 410012, China

[2] College of Information Science and Engineering, Hunan University, Changsha, Hunan 410082, China

[3] National Supercomputing Center in Changsha, Changsha, Hunan 410082, China

[4] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

Published online: 28 November 2018

heuristic scheduling) is a new computing paradigm with good ability to find optima and is a highly effective solution to the NP-hard problem. Many meta-heuristic scheduling algorithms, such as genetic algorithm (GA) (Holland 1992), tabu search algorithm (TS) (Larumbe and Sanso 2013), particle swarm optimization (PSO) (Kennedy and Eberhart 2002), and ant colony optimization (ACO) (Ferrandi et al. 2010), are used in various domains. In recent years, researchers have attempted to solve the task scheduling problem with meta-heuristic scheduling and have made plentiful and substantial achievements. With further studies and tests of guided-random-search-based algorithms, the advantages and disadvantages of each algorithm have been revealed. Gradually, to make better use of the advantages and avoid the shortcomings, researchers have begun to adopt hybrid algorithm strategies.

The BBO algorithm (Simon 2008) created by Simon has attracted attention since 2008. Because of its good optimization ability, BBO has been applied in economics (Bhattacharya and Chattopadhyay 2010a, b), electric power (Rarick et al. 2009; Shafei et al. 2014), control decision and other areas (Lozovyy et al. 2011; Rahmati and Zandieh 2012). Furthermore, various hybrid BBO algorithms have been proposed (Gong et al. 2010; Herbadji et al. 2016; Wang and Xu 2011). The hybrid BBO algorithm is among the most popular in artificial intelligence research areas (Yogesh et al. 2017).

Bharathi et al. (2008) offer researchers who work on planning, scheduling, and execution a wide variety of scientific workflows to evaluate the performance of their implementations. The workflow benchmarks, which can be described as basic workflow structures that are composed into complex workflows by scientific communi ties, are used in this paper, and we use meta-heuristic scheduling to test these benchmarks.

In this paper, we also propose a new hybrid algorithm, which is combined with the heterogeneous earliest-finish-time (HEFT) (Topcuoglu et al. 2002) algorithm, called HMBBO, which formulates the scheduling of directed acyclic graph (DAG) tasks in a heterogeneous cloud computing environment. Moreover, we use the WorkflowSim simulation platform to simulate the performance under the internal benchmark. There are four main contributions:

- Proposal of the implementation of the BBO and PSO algorithms for dependent task scheduling of DAG tasks.
- Hybridization of BBO and PSO to obtain a novel algorithm called HMBBO to produce better solutions and take advantage of the fast convergence rate and high-quality solution.
- Proposal of a new way to combine the classical static scheduling algorithm HEFT with downward static scheduling and analysis of the rank order with test algorithms.
- Assessment of the performance of the algorithms using WorkflowSim and comparison with scientific workflows benchmark. The results show that our algorithm improves the performance.

The remainder of this paper is organized as follows. In section 2, the related work about scheduling algorithms in cloud computing environment is presented. Section 3 describes the system and task model. Section 4 presents the background knowledge of the basic BBO algorithm. Section 5 compares the performance of the proposed algorithm with existing classic heuristic algorithms and analyzes the results. In section 6, the hybrid PSO and BBO scheduling algorithm is presented for DAG task scheduling. The optimization target is to minimize the makespan in a cloud computing environment. Section 7 concludes the paper and notes directions for future work.

2 Related work

In this section, we review recent works relevant to the task scheduling in cloud computing. As a new commercial model, makespan is one of the most important issues in cloud computing, especially for the cloud users. According to whether the task assigned to the cloud computing system is determined, scheduling algorithms can be divided into static scheduling and dynamic scheduling.

Static scheduling refers to the compile stage, where the relationships between tasks are certain. Data dependencies include communication and synchronization requirements between tasks.

In dynamic scheduling, the number of tasks, the data dependencies between tasks, and other factors are uncertain. These factors may change at runtime. However, the overhead of dynamic scheduling is much larger than that of static scheduling. Therefore, cloud computing environments often use static scheduling.

Static task scheduling can be classified into two main groups (Fig. 1), heuristic-based and guided-random-search-based algorithms. The former can be further classified into three groups: list scheduling heuristics, clustering heuristics, and task duplication heuristics (Topcuoglu et al. 2002).

List scheduling heuristics These algorithms maintain a list of all tasks of a specific DAG per their priorities. Every task of the graph is given a priority. Then, a task list is built in decreasing order of priority. Lastly, following the list of precedence given by the graph, the best computational resource is selected for the task with the highest priority, considering a previously defined cost function (Acevedo

et al. 2017). The classic list-based heuristics scheduling algorithms include heterogeneous earliest-finish-time (HEFT) (Topcuoglu et al. 2002), stochastic dynamic level scheduling (SDLS) (Li et al. 2015), predict earliest-finish-time (PEFT) (Arabnejad and Barbosa 2014), and improved predict earliest-finish-time (IPEFT) (Zhou et al. 2017).

Clustering heuristics These algorithms map all tasks with indefinite quantity to different clusters (Boeres and Rebello 2004; Yang and Gerasoulis 1994). In contrast to list scheduling heuristics, clustering heuristics require a second phase to schedule the task clusters to the processors. Only a sufficient number of processors can result in an advantage. The classic clustering heuristics algorithms include dominant sequence clustering (DSC) (Gerasoulis and Yang 1992), task duplication-based scheduling (TDS) (Ranaweera and Agrawal 2000), and heterogeneous selection value (HSV) (Xie et al. 2015).

Task duplication heuristics Task duplication-based heuristics can efficiently reduce the makespan of scheduling DAG tasks when the communication cost is significant, that is, the duplication method reduces or avoids interprocessor communication. Many algorithms have been proposed to incorporate this technique into scheduling (Bansal et al. 2002; Liang and Pang 2016). Bozdag et al. (2009) addressed the problem of schedules that requires a prohibitively large number of processors and proposed a combination of the SDS and SC algorithms to obtain a two-stage scheduling algorithm that produces schedules with high quality and low processor requirements.

Guided random search techniques Guided random search techniques use random choice to guide themselves through the problem space, which is not the same as performing the random walks used in random search methods. These techniques combine the knowledge gained from previous search results with some randomized features to generate new results. The genetic algorithm (Holland 1992; McCall 2005), which was proposed by Holland in 1975, searches for the optimal solution by simulating organic evolution in nature and may be the best known of the guided random search techniques for the task scheduling problem. GA has the advantages of global search ability, simple implementation, and flexibility and has been applied in many fields, including task scheduling (Awadall et al. 2013; Daoud and Kharma 2005; Wang et al. 2002; Xu et al. 2014; Zhang et al. 2015). Like GA, biogeography-based optimization (Simon 2008), particle swarm optimization (Kennedy and Eberhart 2002), and ant colony optimization (Ferrandi et al. 2010) follow similar methods to obtain the optimal solution. Shojafar et al. (2016) proposed a two-phase algorithm for scheduling in cloud computing to decrease energy and makespan. The experiment shows that TETS have an advantage over other two compared algorithms.

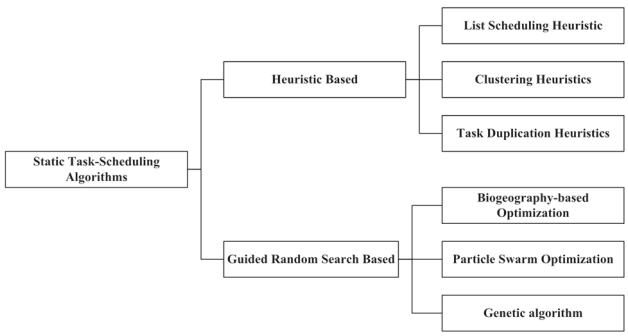


Fig. 1 Classification of static task scheduling algorithms

In this paper, we take the trade-off between makespan and convergence speed into consideration and develop a hybrid approach by integrating BBO with heuristic algorithms.

3 System model

In this section, we first describe the cloud computing structure. Then, we introduce the task model and scheduling model used in this paper.

3.1 Cloud computing model

In cloud computing, users submit tasks to a cloud platform, which allocates these tasks to cloud nodes according to the schedule strategy. Therefore, the schedule strategy is vital for task completion time. Figure 2 illustrates the model of task scheduling in cloud computing. In this paper, tasks are scheduled before the relationships among tasks and the number of task are known. We aim for the optimal distribution strategy to the cloud nodes. Meanwhile, it is assumed that the tasks submitted are non-preemptive. And the task type belongs to DAG task; it means there are dependencies between tasks, when the parent task is not completed, and the child task cannot be executed.

3.2 Task model

Task scheduling models are usually composed of a set of dependent tasks and an interconnected processor. The dependency of the data and the order of execution can be represented by a DAG. Therefore, the DAG scheduling model can be defined as a quaternion (Kopka and Daly 2003):

$$G=(T,E,C,W) \tag{1}$$

where $T=\{t_i|i=1,2,\ldots,n\}$ denotes the collection of n task nodes. $|T|=n$ corresponds to the vertices in the DAG. Each node in the DAG represents a task of a parallel

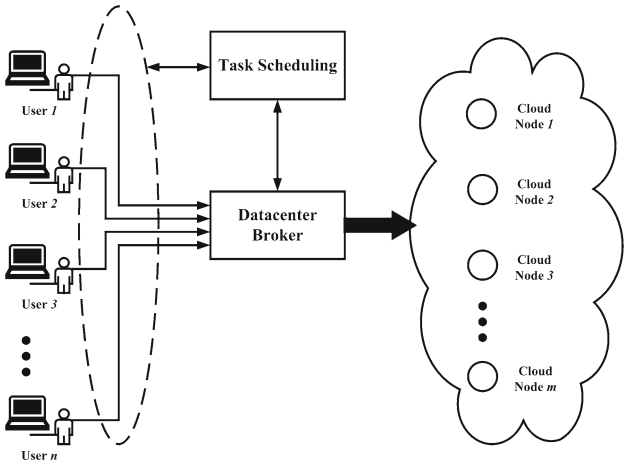


Fig. 2 Model of task scheduling in cloud computing

program. It is usually a code or instruction in the program and is the smallest unit of task scheduling, and it cannot be preempted. The directional edges in the DAG $\{E_{ij} = e(i,j)\} \in E$ denote the relationship between task t_i and t_j. When task t_i is implemented, the results of t_i must be pass to t_j. Therefore, we also call task t_i the predecessor of task t_j (parent node), and task t_j is the successor of task t_i (child node). The weight for the directed edge is defined as which represents the communication cost between task t_i and task t_j. However, a communication cost is only required when two tasks are assigned to different computing nodes. Therefore, the communication cost can be ignored when the tasks are assigned to the same node. The weight on a task t_i is denoted as $w_i \in W$, which represents the computation cost of the task.

Commonly used terms in DAG scheduling are shown in Table 1.

Some of the definitions in DAG scheduling are as follows:

Definition 1 The set of all parent nodes of node t_i is called the parent node set of t_i and is denoted by pred(t_i). The set of all child nodes of node t_i is called the child node set of t_i and is denoted by succ(t_i).

Additionally, if a node does not have any parent nodes, it is called an entry node, denoted by t_{entry}. If a node does not have any child nodes, it is called an exit node, denoted by t_{exit}. If there are multiple entry nodes in a DAG, it is necessary to add a virtual entry node to the DAG. This node has zero cost on each processor, and all real entry nodes are connected to the virtual entry node by a directed edge with zero communication weight. Similarly, if there are multiple exit nodes in a DAG, it is necessary to add a virtual exit node to the DAG. This node has zero cost on each processor, and all real exit nodes are connected to the virtual exit node by a directed edge with zero weight. This approach ensures that the DAG contains only one entry node and only one exit node. It also guarantees the equivalency of the DAG.

Definition 2 When processor p^k has just started a task or completed a task and becomes idle, the time is called the processor idle time, denoted by AT(p^k).

Definition 3 The start time of task t_i on processor p^k is denoted as ST(t_i:p^k):

$$\mathrm{ST}\left(t_i : P^k\right) = \max \left(\max_{t_j \in \mathrm{pred}(t_i)} \left(\mathrm{FT}\left(t_j : p^l\right) + C\left(E_{ji}\right)/W\left(p^l, p^k\right)\right), \mathrm{AT}\left(p^k\right) \right) \tag{2}$$

where $\mathrm{FT}\left(t_j : p^l\right)$ is the completion time of task t_j on processor p^l, $t_j \in \mathrm{pred}(t_i)$. $W\left(p^l, p^k\right)$ represents the communication rate between processor p^l and processor p^k. If $l = k$, then task t_i and its parent task t_j are assigned to the same processor, and the communication cost between them is zero.

Definition 4 The completion time of task t_i on processor p^k is denoted as $\mathrm{FT}\left(t_j : p^k\right)$:

$$\mathrm{FT}\left(t_i : p^k\right) = \mathrm{ST}\left(t_i : p^k\right) + W\left(t_i : p^k\right) \tag{3}$$

Definition 5 The earliest execution time of task t_i on processor p^{k} represents earliest start time, denoted as EST(t_i):

$$\mathrm{EST}\left(t_i, p^j\right) = \max\left\{\mathrm{avail}_j, \max\left(\mathrm{AFT}(t_k) + c\left(t_k, p^j\right)\right)\right\}, \quad t_k \in \mathrm{pred}(t_i) \tag{4}$$

where avail_j represents the earliest time to prepare for task execution of processor p^{j}. AFT(t_k) denotes the actual completion time. EST is calculated from t_{entry}. Thus, the earliest start time of task t_{entry} is $\mathrm{EST}_{\text{entry},\mathrm{p}^j} = 0$.

3.3 Scheduling model

In this paper, the task scheduling problem is defined as mapping a set T of n subtasks in a DAG to a set P of m computing nodes without prejudice to the task priority constraints. Therefore, some relevant parameters in DAG task scheduling, such as EFT, makespan, CCR (communication to computation ratio), etc., can be defined as follows:

Definition 6 The minimum time for task t_i to finish on all processors is called the earliest-finish-time of the task, denoted as EFT(t_i):

Figure 3 shows an example DAG that contains ten tasks. A circle represents a task, and the symbol in a circle

Table 1 Commonly used terms in DAG scheduling

Notation	Definition
$c(i,j)$ or C_{ij}	The communication cost between node i and node j and node j
$e(i,j)$ or E_{ij}	Node i points to a directed edge of node j
t_i	Task node
w_i	The computational cost of node i
CP	Critical path
EFT(t_i)	The earliest completion time of task (t_i)
FT(t_i:p^k)	The completion time of task ti on processor p^k Pk
IL(t_i)	The level of node t_i
N	Total number of nodes
P^k	No. K processor
Pred(t_i)	The predecessor set of node t_i
ST(t_i: p^k)	The start time of task t_i on processor p^k
Succ(t_i)	The successor set of node t_i

represents the number of the task. A directed edge represents the dependency between tasks, and arrows represent the parent–child relationship of tasks. The weight on the edge represents the communication cost between tasks. t_0 is the entry task, and t_9 is the exit task. Table 2 lists the computational cost of each task on three processors (p_1, p_2, p_3). Figure 4 shows the execution schedule of the tasks on all processors using the BBO algorithm.

Definition 7 The makespan refers to the earliest completion time, based on the specific scheduling, of all the tasks on the computation nodes:

$$\text{Makespan} = \max_{i=1}^{n}(\text{EFT}(t_i)) \tag{5}$$

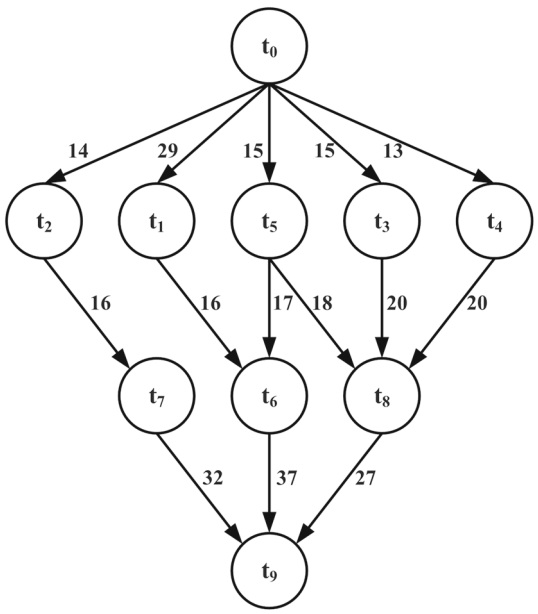


Fig. 3 A simple DAG application model with 10 subtasks

CCR can be used to indicate whether a task graph is computation-intensive or communication-intensive. For a given DAG task graph, CCR is computed as the average communication cost divided by the average computation cost on a target computing system. The computation can be formulated as follows:

$$\text{CCR} = \frac{\sum_{\text{edge}(T_i,T_j)\in E} C(T_i,T_j)}{\sum_{T_i\in T} W(T_i)} \tag{6}$$

Speedup can be defined as follows:

$$\text{Speedup} = \frac{\text{Serial\_execution\_time}}{\text{makespan}} \tag{7}$$

In Eq. 7, the serial execution time is the sum of the average computation times for all tasks, that is, the average computing time on all processors for each task. Thus, the sum of the average completion times for all task is called the serial execution time. In fact, the serial execution time is an approximation in an analog single-core environment. The scheduling length is the use of the corresponding task scheduling algorithm to obtain the execution time, which is called the makespan. Thus, speedup represents the

Table 2 Computing nodes computation speed matrix

t_i	p_1	p_2	p_3
t_0	14	11	17
t_1	13	13	18
t_2	10	13	18
t_3	16	8	17
t_4	12	15	10
t_5	9	16	9
t_6	7	14	11
t_7	5	15	14
t_8	15	12	17
t_9	15	7	16

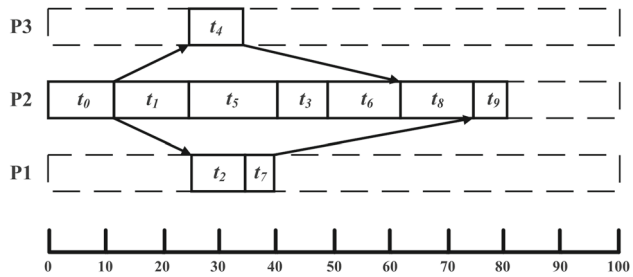


Fig. 4 A simple DAG application model with 10 subtasks

effectiveness of the task scheduling strategy. A higher speedup indicates that the execution time is short and that the scheduling strategy is good.

The efficiency index is the ratio of the speedup to the number of processors and can be used to evaluate the average disposal rate of all processors. The efficiency index can be defined as follows:

$$\text{Efficiency} = \frac{\text{Speedup}}{\text{Number\_of\_processors}} \tag{8}$$

4 BBO algorithm description

In this section, a new heuristic algorithm is developed to solve the DAG task scheduling problem. The BBO algorithm was developed based on the migration of species from one island to another and mutation, in other words, the study of biological population distribution, migration, variation and extinction. The research environment for biogeography is an ecosystem, and there are multiple habitats, as shown in Fig. 5. This algorithm consists of three phases, namely migration, mutation, and clear duplication. In the following subsections, the three phases are presented in detail.

The relation among species, emigration rate, and immigration rate can be found in Fig. 6, where the abscissa axis indicates the number of species and the ordinate indicates the migration ratio. λ represents the rate of species migration, and μ represents the rate of removal of a species. When there are no species in the habitat, the number of species is zero. I indicates that the value of the migration rate is maximized. When the number of species in the habitat reaches the maximum value that the environment can accommodate, the rate of removal reaches the maximum, which is represented as E, indicating that the species is most likely to leave the habitat for a new habitat. When the emigration rate and immigration rate are equal, the number of species reaches S_0 state, and the habitat reaches a state of dynamic equilibrium. When external factors, such as the environment, break this balanced state, the species shift to achieve a new dynamic balance.

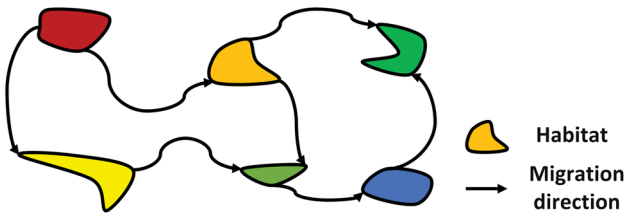


Fig. 5 Ecosystem

The probability that the number of species in a habitat reaches S is denoted as P_s. When the number of habitat species is S, the immigration rate and emigration rate are recorded as λ_s and μ_s. The change in P_s from time t to time $t + \Delta t$ is shown in Eq. 9:

$$P_s(t+\Delta t) = P_s(t)(1-\lambda_s\Delta t - \mu_s\Delta t) + P_{s-1}\lambda_{s-1}\Delta t + P_{s+1}\mu_{s+1}\Delta t \tag{9}$$

To maintain number of species S at the time $t + \Delta t$, a habitat must fulfill three conditions:

1. The number of habitat species at time t is S, and there is no species immigration and emigration at time Δt;
2. The number of habitat species at time t is $S - 1$, and only one species immigrates into the habitat at time Δt;
3. The number of habitat species at time t is $S + 1$, and only one species emigrates from the habitat at time Δt.

If Δt is near zero, immigration and emigration can be ignored. When $\Delta t \to 0$, the limit to Eq. 9 can be found to obtain Eq. 10:

$$P_s' = \begin{cases} -(\lambda_s+\mu_s)P_s + \mu_{s+1}P_{s+1}, S = 0 \\ -(\lambda_s+\mu_s)P_s + \lambda_{s-1}P_{s-1} + \mu_{s+1}P_{s+1}, 1 \le S \le S_{\max}-1 \\ -(\lambda_s+\mu_s)P_s + \lambda_{s-1}P_{s-1}, S = S_{\max} \end{cases} \tag{10}$$

By defining $n = S_{\max}$, $P = [P0 \ldots Pn]T$, Eq. 11 can be represented as $P' = AP$:

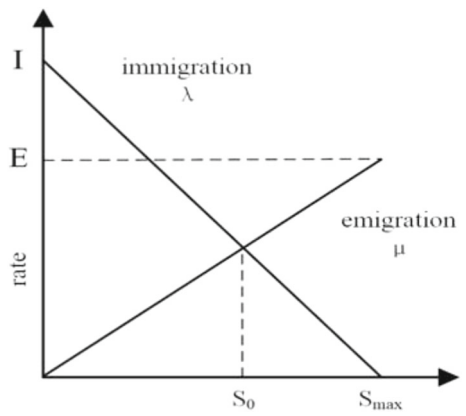


Fig. 6 Species model of a single habitat

$$A=\begin{bmatrix} -(\lambda_0+\mu_0) & \mu_1 & 0 & \cdots & 0 \\ \lambda_0 & -(\lambda_1+\mu_1) & \mu_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \lambda_{n-1} & -(\lambda_{n-1}+\mu_{n-1}) & \mu_n \\ 0 & \cdots & 0 & \lambda_{n-1} & -(\lambda_n+\mu_n) \end{bmatrix} \tag{11}$$

If the number of habitats is k, the emigration rate can be defined as Eq. 12:

$$\mu_k=\frac{Ek}{n} \tag{12}$$

The immigration rate can be defined as Eq. 13:

$$\lambda_k=I\left(1-\frac{k}{n}\right) \tag{13}$$

Suppose $E = I$; then, $\lambda_k+ \mu_k = E$. In this case, we have the A matrix becomes as follows:

$$A=E\begin{bmatrix} -1 & \frac{1}{n} & 0 & \cdots & 0 \\ \frac{n}{n} & -1 & \frac{2}{n} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \frac{2}{n} & -1 & \frac{n}{n} \\ 0 & \cdots & 0 & \frac{1}{n} & -1 \end{bmatrix}=EA' \tag{14}$$

A in Eq. 14 represents the class of birth and death processes in the Markov process. The eigenvalue of matrix A is $[0,-2/n,-4/n,\ldots,-2]$. When the eigenvalue is 0, the corresponding eigenvector is:

$$v=[v_1,v_2,\ldots,v_n] \tag{15}$$

$$v_i=\begin{cases} \dfrac{n!}{(n-1-i)!(i-1)!},\left(i=1,\ldots,\left\lceil\dfrac{n+1}{2}\right\rceil\right) \\ v_{n+2-i},\left(i=\left\lceil\dfrac{n+1}{2}\right\rceil+1,\ldots,n+1\right). \end{cases} \tag{16}$$

where $\lceil n+1/2\rceil$ is the smallest integer greater than $(n + 1/2)$.

Definition 8 When the species in a habitat are in a steady state, the corresponding probability is:

$$P(n)=v\Big/\sum_{i=1}^{n+1}v_i \tag{17}$$

where v and v_i are given in Eqs. 15–16.

4.1 Algorithm flow

The whole algorithm can be divided into three parts: migration, mutation, and clear duplication.

4.1.1 Migration

The BBO algorithm based on the migration model of biogeography described above is used to solve the optimal fitness value. Habitats encounter natural disasters, diseases, and other factors that lead to changes in the environment. Therefore, the number of habitat species can shift out of balance, changing the habitat suitability values. The area that a species lives is defined as the habitat, and the habitat-related parameters include the following:

1. HSI (habitat suitability index)—a parameter that describes the suitability of a habitat for species survival. Higher HSI values indicate better conditions for survival, resulting in more habitat species.
2. SIV (suitable index vector)—HSI-related feature variables form the SIV. Each suitability variable is called an SIV. A habitat with a high HSI tends to have a good solution and many species S.

According to biogeography, a suitable habitat has a more competitive population and inadequate space, so there are more chances to emigrate to nearby habitats. Thus, high HSI habitats have a high emigration rate μ_k Eq. 12 and low immigration rate λ_k Eq. 13, and the opposite is true for habitats with low HSI. The number of species is abstracted as the immigration and emigration probability of each habitat. When the number of habitat populations with higher HSI reaches a value in a saturated state, the migration rate decreases and the removal rate increases. Some species begin to leave the original habitat to find a new habitat. Habitats with lower HSI values increase their values as other species continue to move. Therefore, the habitat HSI value is proportional to the number of biological populations.

4.1.2 Mutation

The BBO algorithm uses a mutation operation to produce this change. The mutation operator causes changes to the simulated habitat due to natural disasters with a certain mutation probability. The mutation rate m is the negative correlation between the number of species. The mutation rate can be obtained as follows:

$$m(S)=m_{\max}\left(\frac{P_{\max}-P_s}{P_{\max}}\right) \tag{18}$$

4.1.3 Flow of the BBO algorithm

The standard flowchart of the BBO algorithm is shown in Fig. 7.

The standard steps in the BBO algorithm are shown in the BBO algorithm's pseudocode:

Algorithm 1 BB0 Algorithm

Require:

Initialize habitats parameters (shown in Table 3)

Ensure:

The best solution and finished

1: Initialize habitat parameters
2: Initialize a random set of habitats that consist of potential solutions to the given problem
3: Calculate the species of each habitat and use for Eq. (12)-(13) to determine the immigration and emigration probability according to the generated habitats
4: **While** (the termination conditions are not satisfied)
5: Update the species count for each habitat
6: Update the immigration and emigration probabilities for each habitat
7: Execute migration operation based on the immigration and emigration probabilities
8: Execute the mutation operation with mutation probability m, given in Eq. (18)
9: **End While**
10: **Return best solution**

4.2 Design principle

According to the attributes of the DAG, a random sequence must satisfy a constraint that maintains a relationship between tasks and subtasks. Note that the initial population of the algorithm must generate sequence sets within the constraint. To ensure efficiency of the initial population, this paper initializes all the sequence solutions in a legal sequence set, as shown in Fig. 8. Therefore, all the populations are legal, and the remaining operations of the algorithm are also legal operations. The next section presents greater details of BBO and PSO.

4.2.1 BBO principle

When using the BBO algorithm, different parameters correspond to various aspects of the problem. Each biological habitat in the BBO algorithm corresponds to a method of solving the problem. Each variable in the habitat corresponds to the dimension of the issue. The suitability of each habitat is calculated by the objective function of each issue. The preferred solution has a high HSI value and therefore a low HSI value. The migration mechanism of a habitat achieves information transfer between methods.

A habitat task scheduling model using the BBO algorithm to solve the task scheduling problem is shown in Fig. 9. In this study, the habitat is modeled with real-number encoding, and this solution vector must satisfy two conditions:

1. Maintain a sequence of relationships between parent tasks and subtasks.
2. All tasks from a DAG must be scheduled.

According to the above statement, a habitat represents a legal sequence and the sequence from DAG. In this paper, the migration operation generates new solutions by exchanging the orders of two tasks. Moreover, the exchanged tasks must be in the legal range, which maintains the relationship between the task and subtask. A task's legal range is the range between the last precursor and the first successor. Therefore, for two tasks to exchange positions, a task must be within another task's legal range. The legal range for migration is illustrated in Fig. 10.

According to the migration strategy, a value in a habitat will perform migration with another allele value in the selected habitat using the roulette method. The nature of the DAG forces the allele values to satisfy the condition that both values are in the legal range of each other. The exchange order operation is a legal operation. The migration operation is illustrated in Fig. 11.

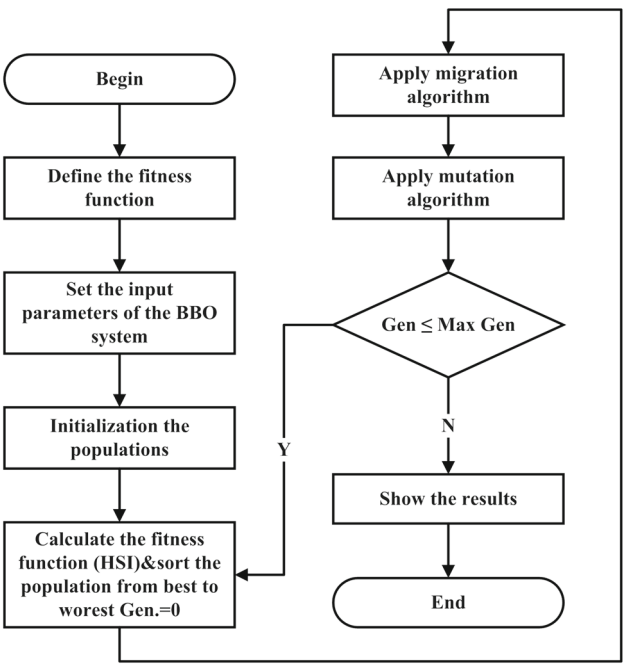


Fig. 7 BBO algorithm flowchart

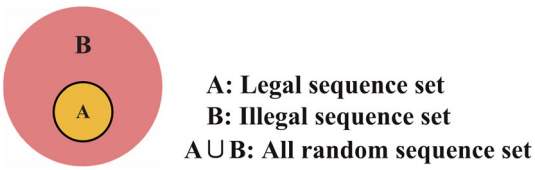


Fig. 8 The all random sequence of a DAG

In this paper, the mutation operation of the BBO algorithm also generates a new legal sequence; the difference with migration is the random method used to generate a new sequence, that is to say, a mutated position (or mutated task) is inserted at a random position to produce a new sequence. The random position must be in the legal range of the mutated task. The mutation process is shown in Fig. 12.

4.2.2 PSO principle

PSO (Kennedy and Eberhart 2002) was created by Kenned and Eberhart based on the foraging of swarm birds and has been applied in practice due to its fast convergence rate, memory function, and simple implementation. In PSO, each particle is a solution vector that exchanges information in flight. In this paper, a particle represents a sequence order of a DAG. The velocity of the particle is defined as the position that a task moves to. The task's legal range in the position update strategy is the same as that in migration. A particle updates its position based on the current task's legal range. The legal range of the flight strategy is shown in Fig. 13.

Specifically, in each generation, every particle updates its movement speed and position according to its personal best and global best to obtain the optimal solution. In this paper, the flight strategy of a particle is designed to update position, that is to say, the VEL matrix represents the positions of all values in a particle that fly. Indeed, the positions of the values represent the execution sequence. Moreover, a flight is invalid if a value in a particle flies to the illegal range of the value, in which case the current value does nothing. The particle velocity equation is given in Eq. 19:

$$v_{id}^{k+1} = \omega \times v_{id}^{k} + c_1 r_1 \left(p_{id}^{k} - x_{id}^{k}\right) + c_2 r_2 \left(g_{id}^{k} - x_{id}^{k}\right) \quad (19)$$

The particle position update equation is given in Eq. 20.

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \quad (20)$$

In Eq. 19, id denotes the dimensionality of a particle, ω is the inertia weight to implement the memory function, k is the current search generation, v is the velocity of the present particle, and c_1 and c_2 are acceleration constants. Furthermore, in Eq. 20, x stands for the current position.

The above rules indicate that the update order operation is a legal operation. The process of self-position updating is shown in Fig. 14.

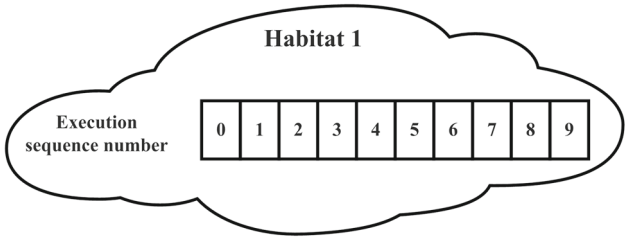


Fig. 9 Model of a habitat

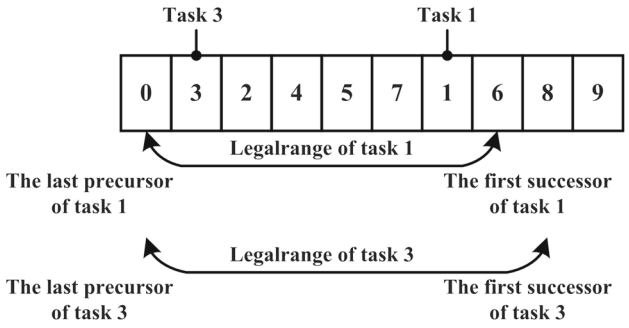


Fig. 10 The legal range for migration

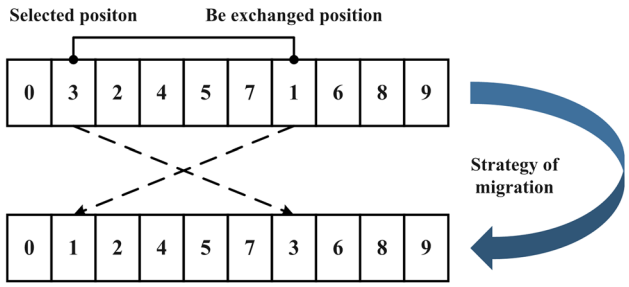


Fig. 11 The process of migration

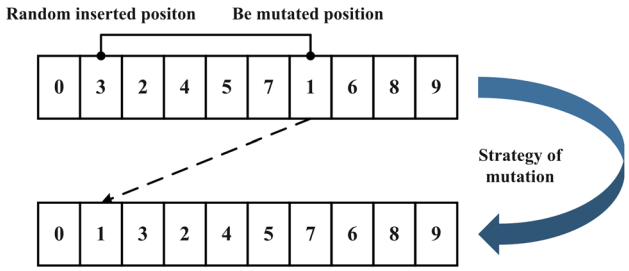


Fig. 12 The process of mutation

5 Experimental results and analysis

In this section, a series of simulation experiment are conducted to verify the performance of BBO algorithm.

In this paper, the steps in all the compared algorithm experiments are divided into sequence-get and processor-allocate, and the minimum EFT of the task allocation strategy is used for all algorithms. The structure of algorithm is based on the HEFT algorithm. In HEFT, the sequence-get part uses the DAG graph attributes to set the

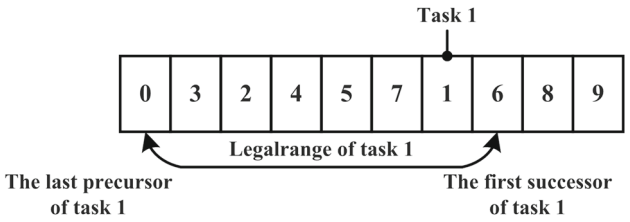


Fig. 13 The legal range for the update strategy

task priorities, which are used to order themselves. The tasks are ordered according to the priorities, which are based on upward and downward ranking. The sequence-get part is the set of legal sequences of the DAG. Moreover, the processor-allocate part uses the same strategy to allocate processors according to each task's EFT as HEFT uses. The best sequence that can be generated by each generation is scheduled in the processor-allocate part. The planning algorithm used to determine the best order is illustrated in Fig. 15.

5.1 Simulation platform

Cloudsim, the could simulate platform that simulates the cloud infrastructure and management service, was developed by Rajkumar Buyya and can be programmed in Java (Calheiros et al. 2009, 2011). Users can obtain the open source program on GitHub. Assigning operations to a cloud simulation platform can save resources and repeated debugging. In this paper, the objective function of the analog simulation experiment is to reach the minimum makespan of task scheduling in cloud computing. Furthermore, Workflowsim (Chen and Deelman 2012) extends the CloudSim simulation toolkit and supports a multilayered model of failures and delays occurring in the various levels of the Workflow management system.

Figure 16 shows the components of Workflowsim involved in preparing and executing a workflow. All the processing procedures form a workflow management system (WMS), which is similar to that of the Pegasus WMS (Deelman et al. 2005, 2015). In Fig. 16, the components are the Workflow Mapper, Clustering Engine, Workflow Engine, and Workflow Scheduler.

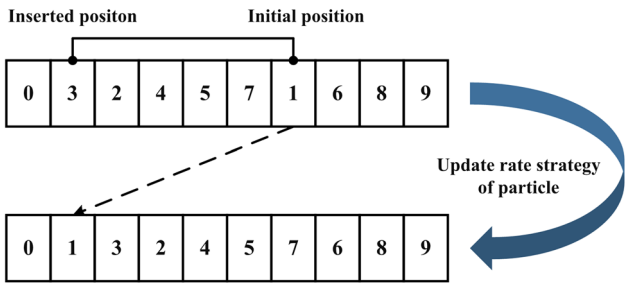


Fig. 14 The particle update strategy

A Workflow Mapper parses imported DAG files formatted in XML. The Workflow Engine is used to handle the data dependencies, and the Workflow Scheduler matches tasks to resources. The Clustering Engine merges similar types of tasks into a massive job to improve the execution efficiency.

The new features of WorkflowSim are the Failure Generator, Failure feedback system and Overhead of delays at different levels, which enable a more accurate simulation of a real cloud computing environment. Some necessary measures are fixed in WorkflowSim to support static scheduling. First, the execution sequence generated by HEFT is changed by the WMS because WMS structure is suitable for dynamic scheduling. In WorkflowSim, the WMS processes the DAG tasks according to the ScheduledList submitted by the scheduling method. The dynamic schedulers are included in the package of org.workflowsim.scheduling and submit the execution order of the DAG tasks to ScheduledList. However, the planning

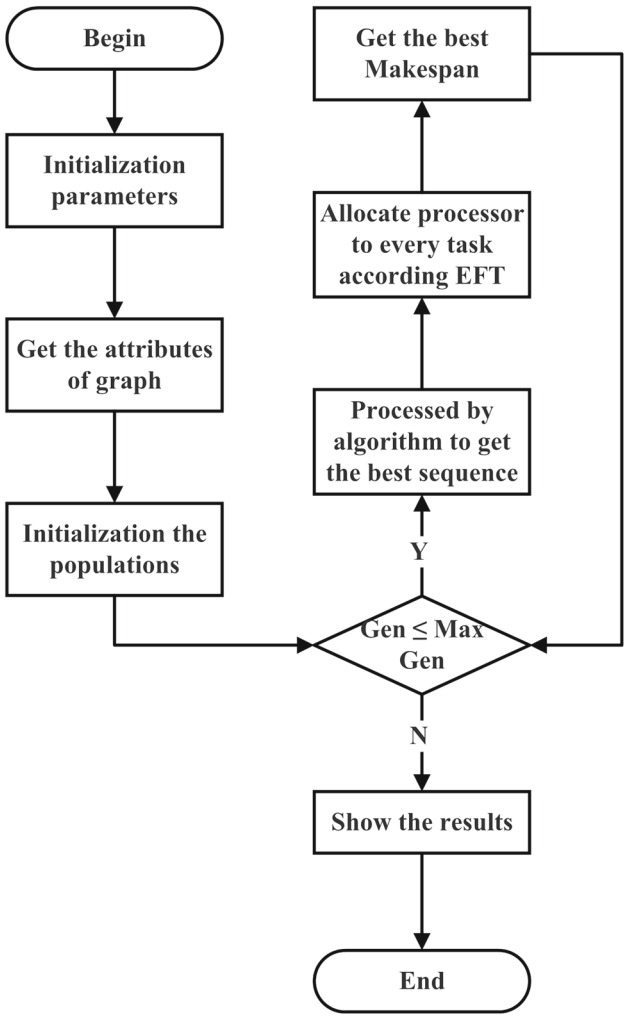


Fig. 15 Planning algorithm to determine the best order

algorithms do not submit the order to ScheduledList directly. Furthermore, the task execution order is planned by the planning algorithm, which indirectly considers the state of CondorVM, that is, the program will not pass WMS. Therefore, we extend only the modules in the org.workflowsim.scheduling package that contain a objective function, processor allocation, validity check and so on. Moreover, the set of HEFT is used to prove that the program works well, and the results of HEFT planning are shown in Fig. 17.

5.2 DAG benchmark

The planning and scheduling experiments require access to multiple types of scientific workflows to evaluate the performance (Xu et al. 2013). Four diverse scientific applications introduced in Xu et al. (2013) are used for the DAG benchmark to evaluate the performance. The DAG benchmarks include the following: Montage (Montage: An astronomical image engine 2006), which was created by NASA/IPAC Infrared Science Archive as an open source toolkit. This toolkit can be used to generate custom mosaics of the sky using input images in flexible image transport system (FITS) format; CyberShake (Deelman et al. 2006) which is used by the Southern California Earthquake Center (SCEC) to characterize earthquake hazards in a region using the probabilistic seismic hazard analysis (PSHA) technique; Laser Interferometer Gravitational-Wave Observatory (LIGO) Inspiral Analysis Workflow (Brown et al. 2007), which is used to analyze the data obtained from coalescing compact binary systems, such as binary neutron stars and black holes; and the *s*RNA Identification Protocol using High-throughput Technology (SIPHT) program (Livny et al. 2008), which uses a workflow to automate the search for sRNA-encoding genes for all the bacterial replicons in the National Center for Biotechnology Information (NCBI) database. More information about the workflows is given in the workflow gallery (Workflow gallery 2018), which categorizes the workflows run by an application.

All the benchmarks can be generated with the Synthetic Workflow Generators program, which can be used to evaluate the scheduling and provisioning algorithms for scientific workflow management systems available at (Workflow Generator 2006). The workflow generators (Bharathi et al. 2008; Da Silva et al. 2014; Juve et al. 2013) are based on models of real applications that have been parameterized with file size and task runtime data from execution logs and publications that describe the workflows.

5.3 Benchmark experiments and results

The experiment compares the Makespan, convergence rate and efficiency of BBO, PSO, and HEFT with static downward and upward ranks (Topcuoglu et al. 2002). For better performance than heuristic algorithms, HEFT is selected as the experimental subject (Xu et al. 2013). We choose PSO as a comparison algorithm because of its fast convergence rate, wide use, memory function, and simple implementation. The BBO and PSO parameter settings are described in Tables 3 and 4. The parameters are the settings that result in the best performance in the current environment.

In this experiment, the settings of the simulation platform WorkflowSim have no overhead and no clustering. All the simulations are tested in the same software environment with the same objective function, results read function, and index settings. The bandwidth environment is set to be homogenous to simplify the experimental settings, i.e., the bandwidth is the same for all processors. Each virtual machine has a different million instructions per second (MIPS) range from 500 to 3600. With the sets of WorkflowSim, the transfer cost between a parent and child is calculated as the size of the input type file/bandwidth, and the computation cost of a task on a virtual machine is

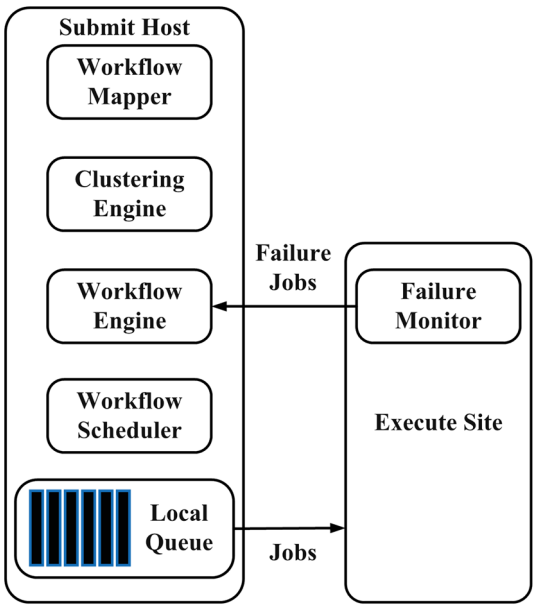


Fig. 16 Structure of WorkflowSim

========== OUTPUT ==========

Job ID	Task ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time	Depth
10	Stage-in	SUCCESS	2	0	0.22	0.1	0.32	0
0	1,	SUCCESS	2	3	17.5	0.32	17.82	1
1	4,	SUCCESS	2	3	16.25	17.82	34.07	2
2	2,	SUCCESS	2	2	18.571429	17.820036	36.391465	2
3	5,	SUCCESS	2	1	20	17.820022	37.820022	2
4	3,	SUCCESS	2	0	22	17.820024	39.820024	2
5	6,	SUCCESS	2	3	16.25	34.07	50.32	2
6	9,	SUCCESS	2	2	25.714286	37.820048	63.534334	3
7	7,	SUCCESS	2	1	11.666667	39.820064	51.486731	3
8	8,	SUCCESS	2	3	6.25	50.32	56.57	3
9	10,	SUCCESS	2	3	26.25	63.53436	89.78436	4

Fig. 17 The results of HEFT planning

computed as the runtime of the task MIPS of the virtual machine.

First, owing to its simple structure and accessibility, we use the Inspiral DAG benchmark with 100 tasks. Inspiral has a simple structure, with the shape shown in Fig. 18. In the first experiment, Inspiral is tested on different numbers of virtual machines (4, 8, 16, and 32) with a bandwidth of 500 Mb/s. We run each program ten times and take the average. The maximum number of iterations for each heuristic algorithm is 2000. Clearly, HEFT with downward rank and upward rank runs only one time. For convenience, the downward and upward rank methods of HEFT are, respectively, abbreviated as HEFT_D and HEFT_U.

Figure 19 shows the makespan of the four algorithms for different numbers of virtual machines. The makespan shows a decreasing trend with increasing number of virtual machines. BBO has the advantage among the four algorithms with increasing number of virtual machines. HEFT_D performs well when there are more than 4 virtual machines, which demonstrates the validity of the classic algorithm. By contrast, PSO and HEFT_U have relatively poor performance due to a tendency to become trapped in local minima as the number of virtual machines increases.

A comparison of the efficiency is shown in Fig. 20. The efficiency with 32 virtual machines for 100 tasks is worse in the parallel environment. Therefore, increasing the number of processors may not be beneficial. As the figure shows, BBO and HEFT_D have better performance because as the number of virtual machines increases, the handling capacity increases, but the advantage of parallelism decreases.

Figures 21, 22, 23 and 24 compare the convergence rates of the four algorithms with 4, 8, 16 and 32 virtual machines. BBO outperform HEFT_D with 4 and 8 virtual machines, but its performance is poor under the other conditions. Except with 16 virtual machines, BBO converges at approximately the 500th generation. By contrast, PSO prematurely converges in all situations in fewer generations than BBO. Because BBO has excellent global search ability and jumps out of local optima, the algorithm produces better results when run on 4 and 8 virtual machines. However, as the number of virtual machines increases, the convergence rate slows and the results become undesirable because the search space increases. As a result, the greedy strategy (HEHT) produces better performance.

The above experiments show that HEFT_D still achieves better results than those of the other algorithms

Table 3 BBO parameters

Parameters	Value
Number of habitats	50
Maximum mutation probability $m_{\max}$	1
Number o f elitism habitats	2
Maximum immigration rate I	1
Maximum emigration rate E	1

Table 4 PSO parameters

Parameters	Value
Number of particles	50
Number o f elitism particles	2
Inertia weight ω	0.2
Acceleration constant c_1	0.9
Acceleration constant c_2	0.9

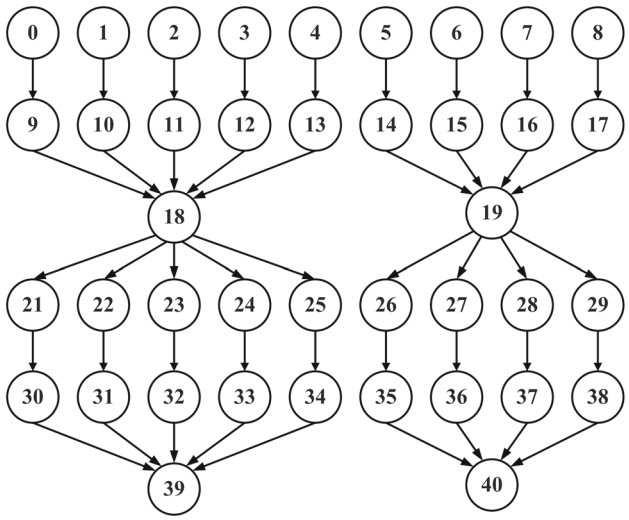


Fig. 18 Structure of Inspiral

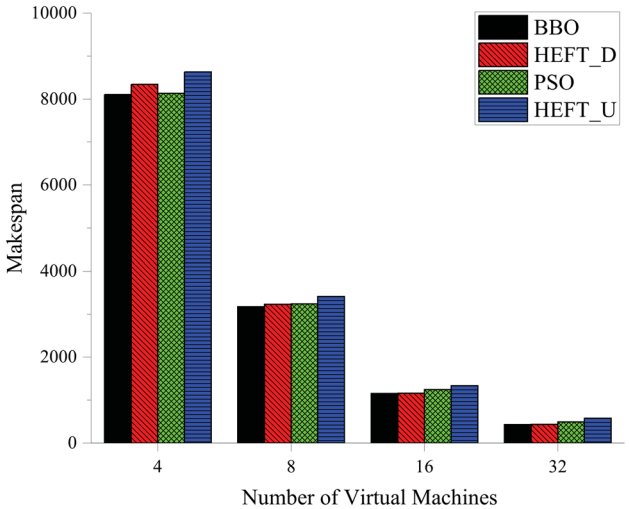


Fig. 19 Makespan of the four algorithms with different numbers of virtual machines

and that setting the priority of each task in HEFT_D is still a useful method to evaluate the sequence of tasks. By contrast, BBO performs well in some situations, but the results of other algorithms are better after too many generations. The premature termination of PSO in the search process results in relatively poor solutions. The reason PSO becomes trapped in local optima is that its memory function cannot adapt to obtain the sequence of tasks. Additionally, even though BBO is an excellent algorithm, it does not work well in all situations. To overcome these problems, hybrid migration BBO (HMBBO) is proposed in the next section.

6 HMBBO algorithm

In this section, a novel hybrid algorithm has been proposed. A series of simulation experiment and real experiment are conduced to verify the performance of our algorithm.

In the first experiment, the HEFT_D algorithm produces excellent results, and the PSO algorithm has a better convergence rate. Although BBO does not produce good results or have a fast convergence rate, its makespan is the best. To improve the performance, we develop a hybrid of the flight strategy under the BBO migration structure to accelerate the search speed of BBO. Moreover, we use HEFT_D to evaluate whether a task sequence is good. That is to say, according to the downward method in HEFT, the execution order of each task is based on its downward order. The more significant the rank of a task is, the earlier the task will be executed. The evaluation of a sequence as good or bad based on an objective function method in each migration consumes too much time, i.e., the process used to allocate tasks to virtual machines according to a new

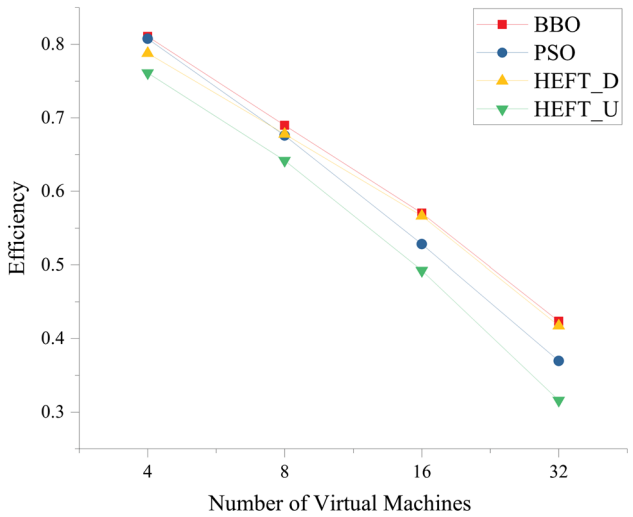


Fig. 20 Efficiency comparison for the Inspiral

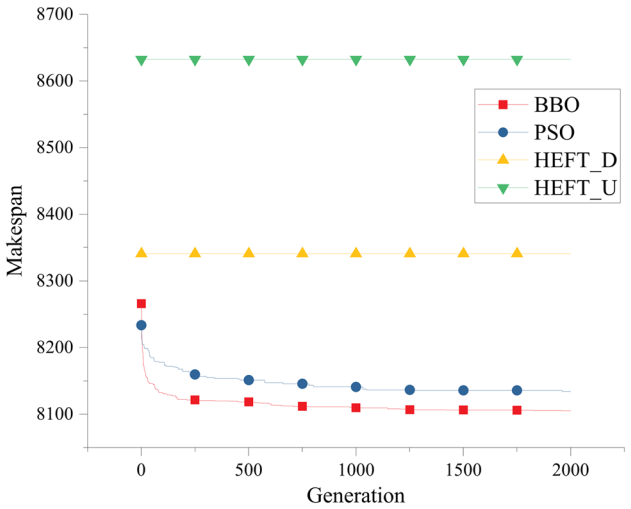


Fig. 21 Comparison of the convergence rate for 4 virtual machines

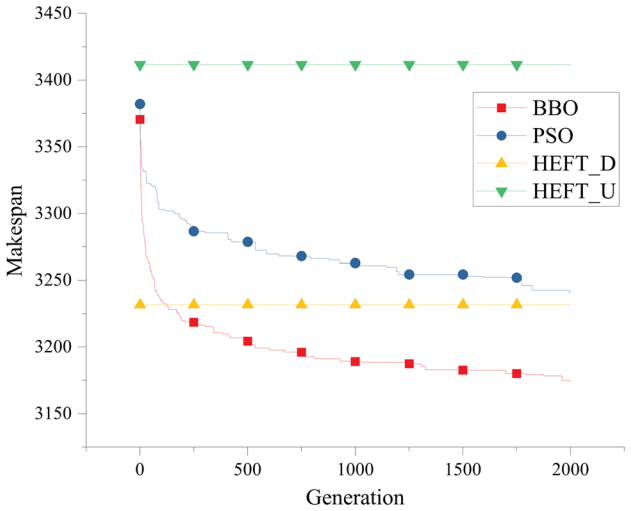


Fig. 22 Comparison of the convergence rate for 8 virtual machines

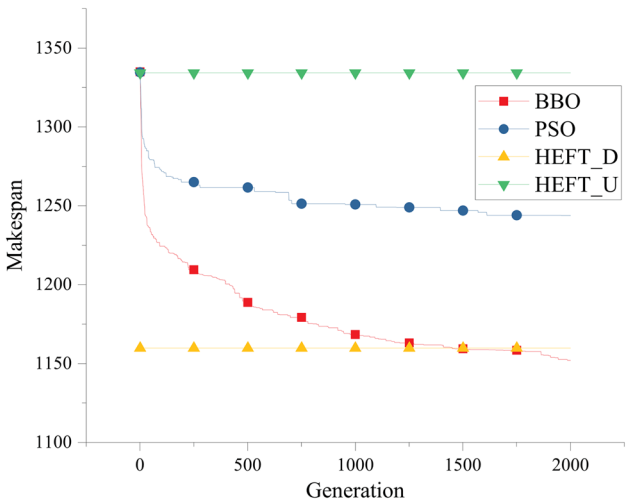


Fig. 23 Comparison of the convergence rate for 16 virtual machines

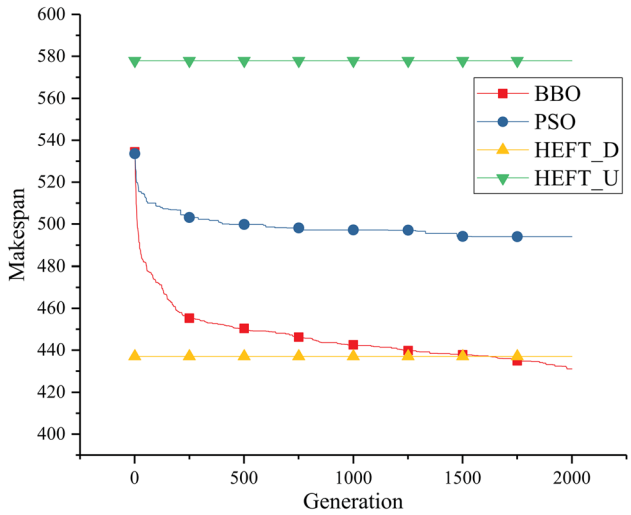


Fig. 24 Comparison of the convergence rate for 32 virtual machines

sequence has high time complexity. Therefore, Eq. 21 could be used to evaluate whether a sequence is good. According to the proposed method, the tasks' rank of a DAG is calculated one time, and then, each task receives a characteristic rank.

$$S_{\text{rank}} = \sum_{i=0}^{\text{TaskNum}-1} ((\text{TaskNum} - i) \times \text{rank}_i)^2 \tag{21}$$

In Eq. 21, S_{rank} denotes the rank of a sequence. TaskNum is the task number of a sequence, and rank_i denotes the task's rank based on the downward method, which is precalculated. We use $\text{TaskNum} - i$ as the precedence function to weight the tasks in a sequence.

The flow of HMBBO algorithm is generally same as BBO algorithm in initialization and mutation part. The cross operation part is the most different; for save space, this paper only gives the migration part. The detail of HMBBO algorithm's hybrid migration strategy is given in pseudocode in Alg. 2.

In Alg. 2, it cloud be found that the time complexity in cross operation part mainly comes from the calculation of Eq. 21. Calculating Eq. 21 costs is O(TaskNum); therefore, the time complexity of HMBBO cross operation part is O(*complexity of BBO cross operation* + TaskNum). From the whole flow of HMBBO, the main time complexity is the calculating of cross operation part. In this paper, consider the time complexity of BBO and PSO are almost the same, and then, the time complexity of HMBBO is the same to BBO because every generation only execute either BBO migration or PSO flight. Besides, the BBO habitats and PSO particles have common structure and could share the same space in program, so the space complexity have no increase compare to BBO.

Algorithm 2 Hybrid migration strategy

Require:
Habitat x
Ensure:
The new habitat y
1: Select Value A of x by roulette selection to be operated on
2: **if** A satisfies the legal range of BBO&PSO **then**
3: Get the maximum rank value x_r using Eq. (21) for the three situations that contains the strategy BBO, PSO and original sequence of x
4: **switch** x_r
5: **case**: x_r = rank value of BBO
6: Generate new habitat B with migration
7: **case**: x_r = rank value of PSO
8: Generate new habitat B with update rate strategy
9: **else if** A only satisfies the legal range of BBO, then
10: Generate new sequence with migration
11: **else if** A only satisfies the legal range of PSO, then
12: Generate new sequence with update rate strategy
13: **end if**
14: **return** B

6.1 Benchmark DAG experiment

The second experiment is the same as the first but is based on the makespan, efficiency, and convergence rate. All the results are shown in Figs. 25, 26, 27, 28, 29, and 30.

As in the first experiment, the benchmarks include 100 tasks, except SIPHT with 97 tasks, and the results are shown in Figs. 31, 32, and 33.

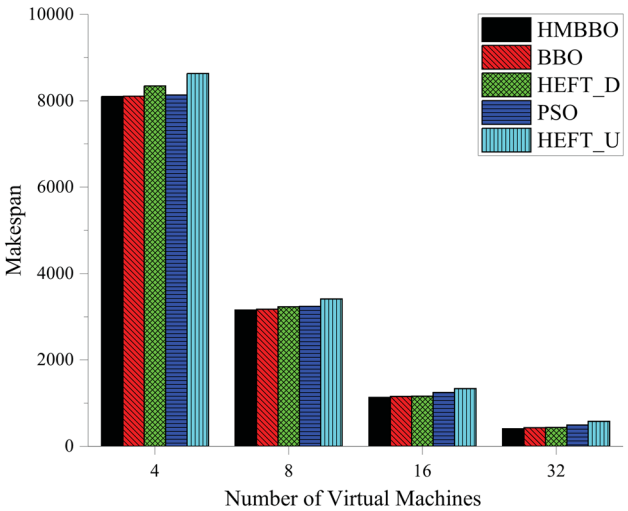


Fig. 25 Makespan of the five algorithms with different numbers of virtual machines

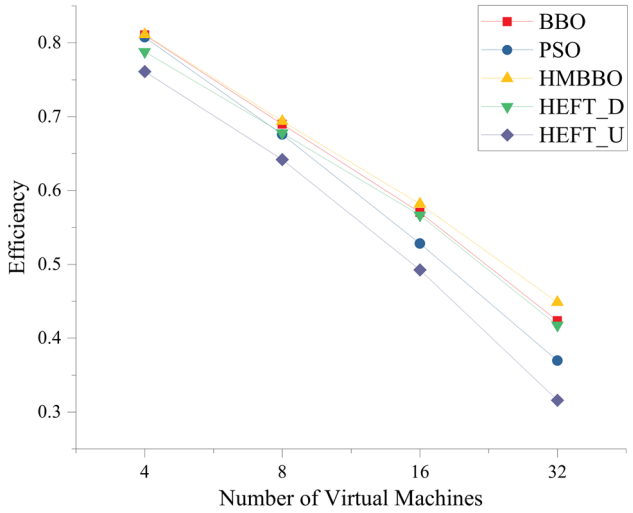


Fig. 26 Efficiency comparison for Inspiral

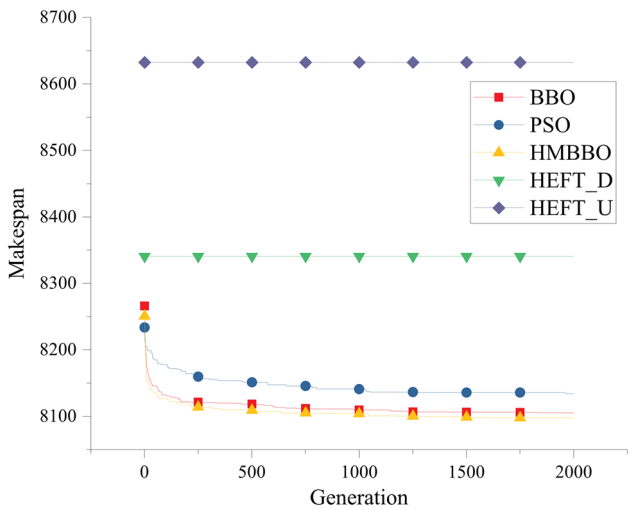


Fig. 27 Comparison of the convergence rate for 4 virtual machines

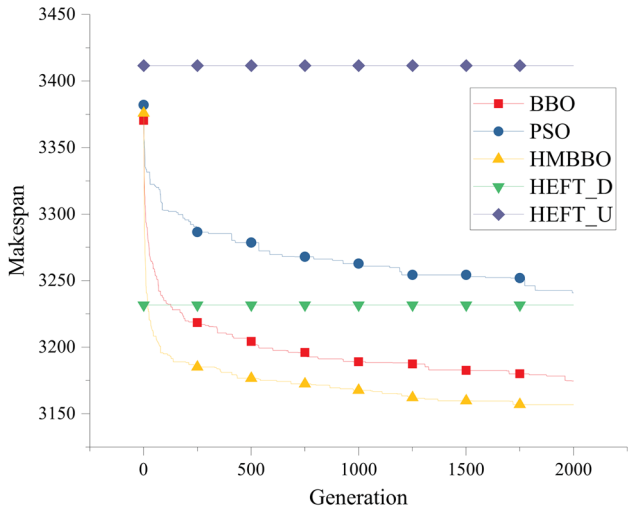


Fig. 28 Comparison of the convergence rate for 8 virtual machines

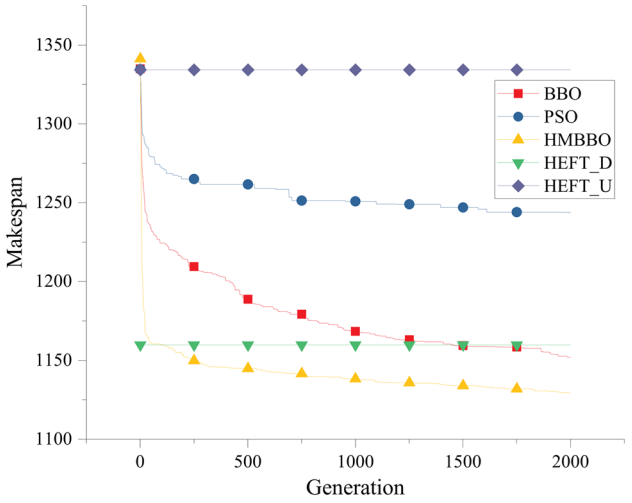


Fig. 29 Comparison of the convergence rate for 16 virtual machines

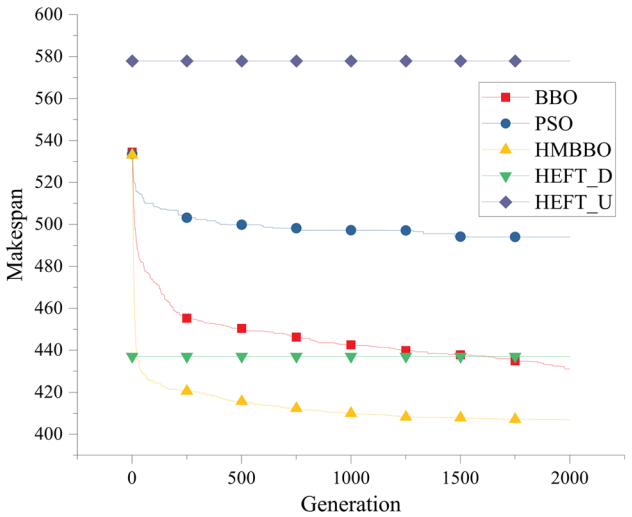


Fig. 30 Comparison of the convergence rate for 32 virtual machines

HMBBO outperforms the other methods on all four benchmarks. Because the structure of each graph is different in the CCR environment, some of the DAG benchmarks, such as Montage, CyberShake, and Sipht, result in near-optimum solutions with our algorithms. In general, the performance of HMBBO is better than that of the other algorithms on all DAG benchmarks.

According to the results of the second experiment, HMBBO inherits the strong ability for optimality from BBO and has the advantages of simple implementation, excellent performance, and fast convergence rate. Moreover, the makespan results of HMBBO are better than those of HEFT_D. The most obvious advantage is the faster convergence rate, which requires fewer generations to surpass the results of the other algorithms. The results show that the strategy achieves the desired goal and solves

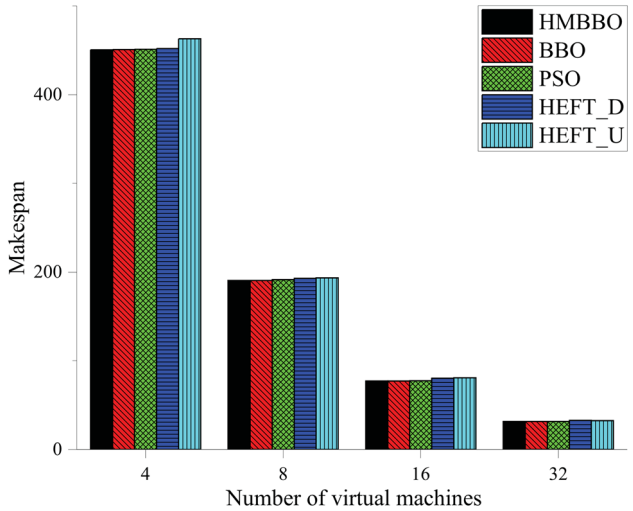


Fig. 31 Makespan of five algorithms with different numbers of virtual machines using Montage

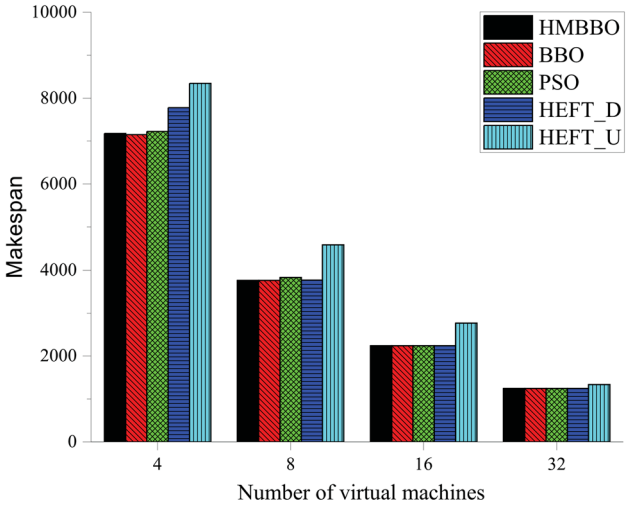


Fig. 33 Makespan of five algorithms with different numbers of virtual machines using Sipht

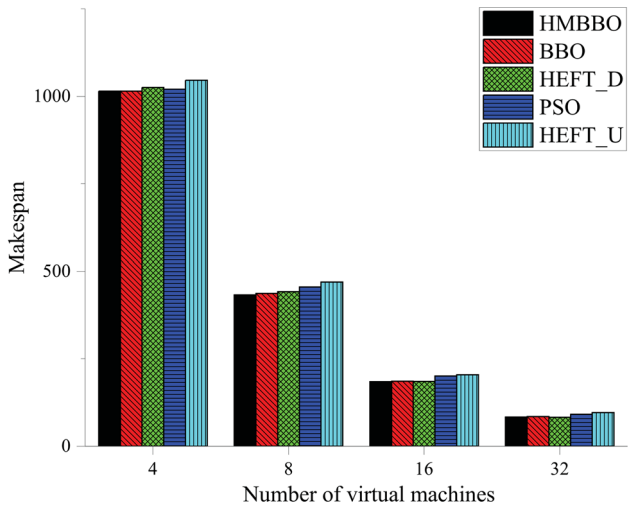


Fig. 32 Makespan of five algorithms with different numbers of virtual machines using CyberShake

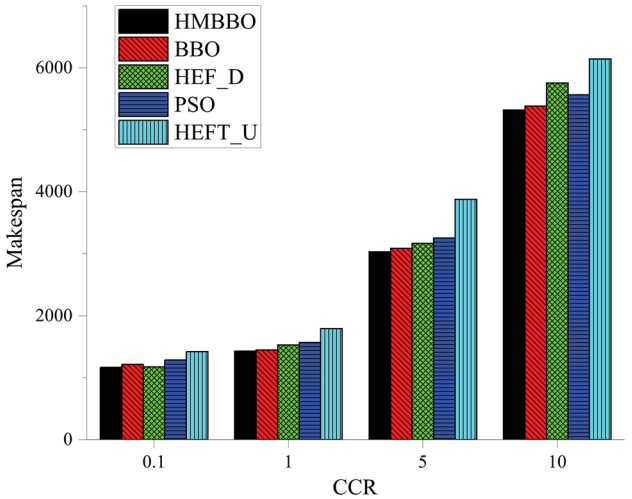


Fig. 34 Makespan of five algorithms under different CCR values using Inspiral with 100 tasks; the number of virtual machines is 16

similar problems. The hybrid migration strategy balances exploration and greed effectively, combining the advantages of the good solutions of greedy strategies and the powerful search ability of HEFT, BBO, and PSO.

In the third experiment, to further test the performance of the algorithms under different heterogeneous conditions, it is necessary to test different CCR values. Figure 34 shows the results of all the algorithms run on the Inspiral benchmark with 100 tasks under different CCR values. As the transfer costs increase, the makespan also increases. HMBBO still has the advantage in all environments, and HEFT_D produces poorer results with increasing CCR values. Additionally, HEFT becomes less efficient in communication-intensive applications. By contrast, the other algorithms produce better results due to their wide searches.

In the fourth experiment, we set the CCR value to 0.1, 1, 5, and 10 and use the DAG benchmarks Montage, CyberShake, Inspiral, and Sipht to test the performance under different environments. The number of virtual machines is 8 or 32. Figures 35 and 36 show that HMBBO, BBO, and PSO produce relatively optimal solutions, but the two algorithms of HEFT have poor performance because HEFT cannot adapt to every benchmark and the other meta-heuristic scheduling algorithms produce relatively optimal solutions. The results suggest that the HEFT algorithm is limited by the different structures of the benchmarks. By contrast, meta-heuristic scheduling has a broader application due to its strong and general search ability.

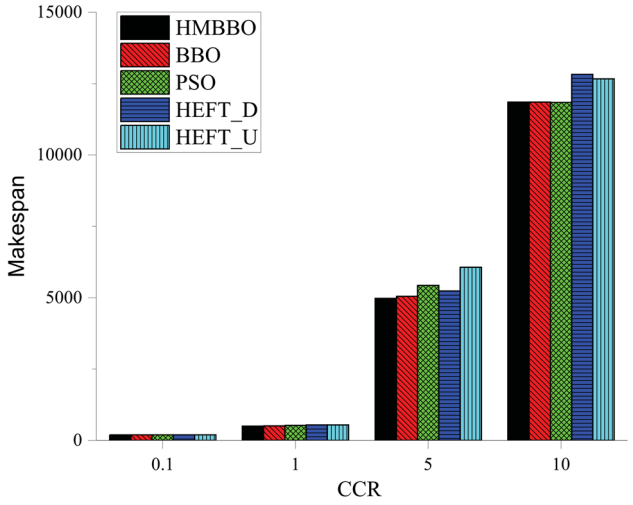


Fig. 35 Makespan of five algorithms under different CCR values; the number of virtual machines is 8

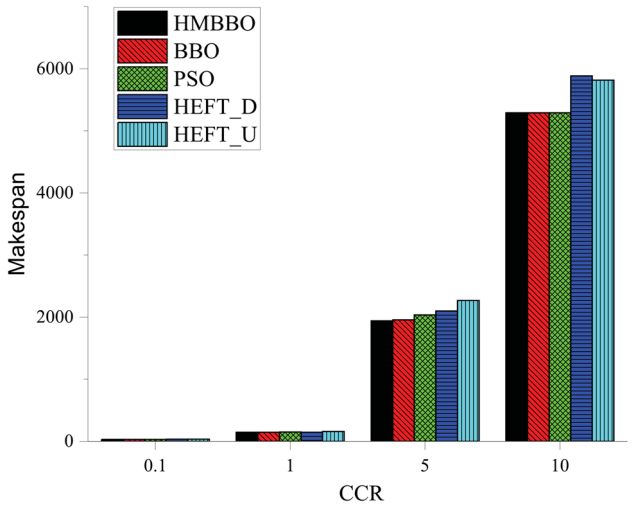


Fig. 36 Makespan of five algorithms under different CCR values; the number of virtual machines is 32

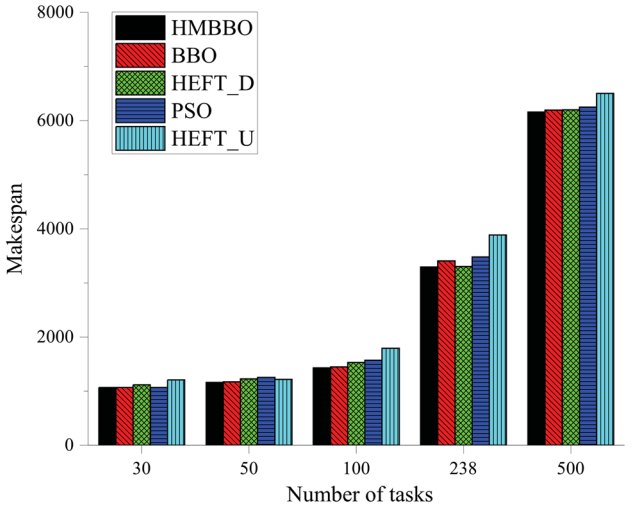


Fig. 37 Makespan of different tasks using Inspiral, CCR = 1, and 16 virtual machines

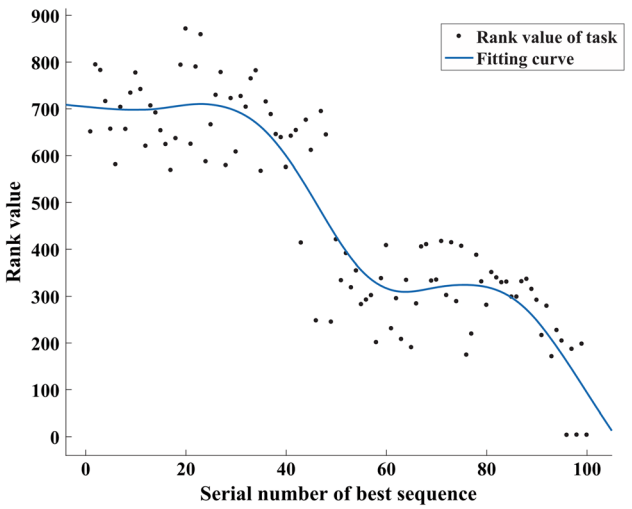


Fig. 38 Rank value of best sequence with BBO

In the fifth experiment, we test the influence of a different number of tasks for all algorithms. We vary the number of tasks from 30 to 500. In one case, the number of tasks is 238 because the workflow generator for Inspiral with 238 is an integrity structure. The results in Fig. 37 show that as the number of tasks increases, the makespan also increases. Furthermore, BBO and HEFT_D alternately outperform each other, and HMBBO always produces the best results. Therefore, the hybrid migration strategy can adapt to different environments because of its strong random-search ability.

Lastly, we use Matlab to fit the rank value of the final sequences generated by BBO, HMBBO, and HEFT_D with the toolkit smoothing spline with a smoothing parameter of 0.00067321295. The fitting curves are illustrated in Figs. 38, 39, and 40, which show the tendencies of the best sequences for the different algorithms. HEFT_D follows the downward order of rank value. Owing to the HEFT_D strategy, the distribution of the random value in HMBBO is more concentrated than that of BBO, and the general trend is more similar to that of HEFT_D. The results illustrate that in guided-random-search-based algorithms, applying a positive influence will improve the results of the entire evolutionary system. Furthermore, in the task scheduling problem, if we implement HEFT_D with the greedy algorithm, maintaining an appropriate distance to HEFT_D would result in better solutions.

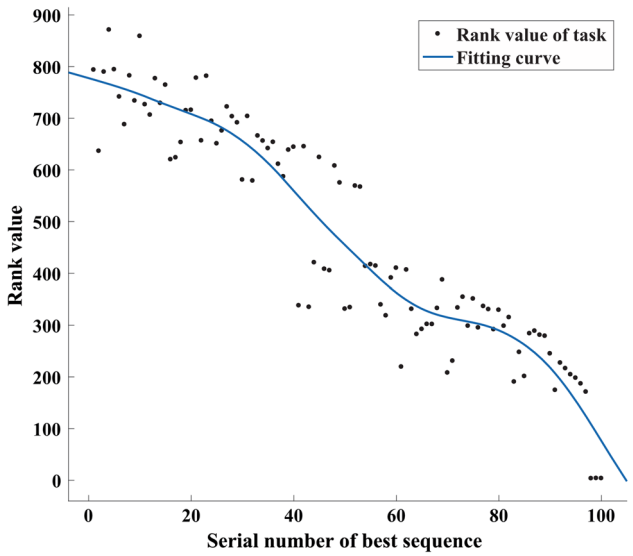


Fig. 39 Rank value of best sequence with HMBBO

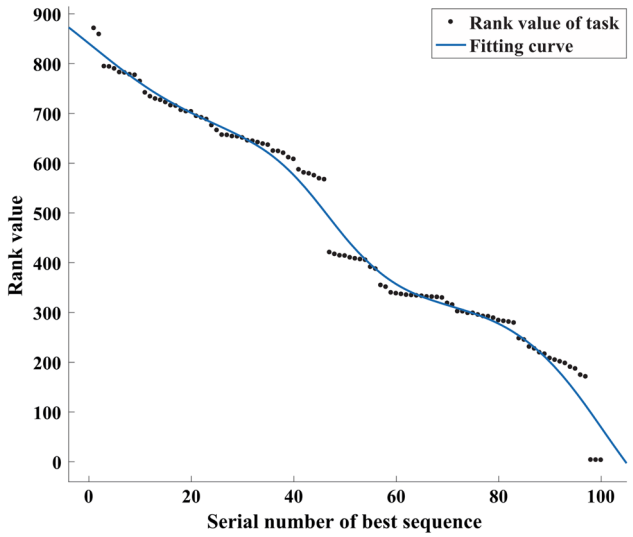


Fig. 40 Rank value of best sequence with HEFT D

7 Summary and future work

In this paper, we integrate a migration strategy with BBO and PSO to solve the problem of scheduling DAG tasks in a cloud computing environment. According to the experimental results, HMBBO inherits the strong ability for optimality from BBO and has the advantages of simple implementation, excellent performance and fast convergence rate with different benchmarks compared with BBO, PSO, and HEFT, which proves that HMBBO can be applied to task scheduling in cloud computing environments. Additionally, we propose a new way to combine the method with the classic static algorithm HEFT and test these algorithms on WorkflowSim run with scientific workflow benchmarks.

In the future, we intend to propose a new method of reinforcement learning to extend our algorithm to further improve the results. Due to the development of cloud computing, the number of tasks is increasing and data environment has become complicated. The most important point is the rapid development of artificial intelligence, deep machine learning, and neural network engines to make the process of optimum searching more intelligent. Therefore, the addition of intelligence to our algorithm is inevitable.

Acknowledgements The research was partially funded by the Program of National Natural Science Foundation of China (Grant No. 61502165), the National Outstanding Youth Science Program of National Natural Science Foundation of China (Grant No. 61625202).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Acevedo C, Hernández P, Espinosa A, Méndez V (2017) A critical path file location (CPFL) algorithm for data-aware multiworkflow scheduling on HPC clusters. Future Gener Comput Syst 74:51–62

Arabnejad H, Barbosa JG (2014) List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Trans Parallel Distrib Syst 25:682–694

Awadall M, Ahmad A, Al-Busaidi S (2013) Min-min ga based task scheduling in multiprocessor systems. Int J Eng Adv Technol 2:2249–8958

Bansal S, Kumar P, Singh K (2002) Duplication-based scheduling algorithm for interconnection-constrained distributed memory machines. In: International conference on high-performance computing. Springer, pp 52–62

Bharathi S, Chervenak A, Deelman E, Mehta G, Su M-H, Vahi K (2008) Characterization of scientific workflows. In: 3rd Workshop on workflows in support of large-scale science. WORKS 2008. IEEE, pp 1–10

Bhattacharya A, Chattopadhyay PK (2010a) Biogeography-based optimization for different economic load dispatch problems. IEEE Trans Power Syst 25:1064–1077

Bhattacharya A, Chattopadhyay PK (2010b) Solving complex economic load dispatch problems using biogeography-based optimization. Expert Syst Appl 37:3605–3615

Boeres C, Rebello VE (2004) A cluster-based strategy for scheduling task on heterogeneous processors. In: 16th symposium on computer architecture and high performance computing. SBAC-PAD 2004. IEEE, pp 214–221

Bozdag D, Ozguner F, Catalyurek UV (2009) Compaction of schedules and a two-stage approach for duplication-based DAG scheduling. IEEE Trans Parallel Distrib Syst 20:857–871

Brown DA, Brady PR, Dietz A, Cao J, Johnson B, McNabb J (2007) A case study on the use of workflow technologies for scientific analysis: gravitational wave data analysis. In: Workflows for e-Science. Springer, pp 39–59

Calheiros RN, Ranjan R, De Rose CA, Buyya R (2009) Cloudsim: a novel framework for modeling and simulation of cloud computing infrastructures and services arXiv preprint arXiv:09032525

Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41:23–50

Chen W, Deelman E (2012) Workflowsim: a toolkit for simulating scientific workflows in distributed environments. In: 2012 IEEE 8th international conference on E-science (e-science). IEEE, pp 1–8

Da Silva RF, Chen W, Juve G, Vahi K, Deelman E (2014) Community resources for enabling research in distributed scientific workflows. In: 2014 IEEE 10th international conference on e-Science (e-Science). IEEE, pp 177–184

Daoud M, Kharma N (2005) Gats 1.0: a novel ga-based scheduling algorithm for task scheduling on heterogeneous processor nets. In: Proceedings of the 7th annual conference on genetic and evolutionary computation. ACM, pp 2209–2210

Deelman E et al (2005) Pegasus: a framework for mapping complex scientific workflows onto distributed systems. Sci Program 13:219–237

Deelman E et al (2006) Managing large-scale workflow execution from resource provisioning to provenance tracking: the cybershake example. In: 2nd IEEE international conference on e-Science and grid computing. e-Science'06. IEEE, pp 14

Deelman E et al (2015) Pegasus, a workflow management system for science automation. Future Gener Comput Syst 46:17–35

Ferrandi F, Lanzi PL, Pilato C, Sciuto D, Tumeo A (2010) Ant colony heuristic for mapping and scheduling tasks and communications on heterogeneous embedded systems. IEEE Trans Comput Aided Des Integr Circuits Syst 29:911–924

Gerasoulis A, Yang T (1992) A comparison of clustering heuristics for scheduling directed acyclic graphs on multiprocessors. J Parallel Distrib Comput 16:276–291

Gong W, Cai Z, Ling CX (2010) DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization. Soft Comput 15:645–665

Herbadji O, Slimani L, Bouktir T (2016) Solving bi-objective optimal power flow using hybrid method of biogeography-based optimization and differential evolution algorithm: a case study of the Algerian electrical network. J Electr Syst 12:197–215

Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, Cambridge

Juve G, Chervenak A, Deelman E, Bharathi S, Mehta G, Vahi K (2013) Characterizing and profiling scientific workflows. Future Gener Comput Syst 29:682–692

Kennedy J, Eberhart R (2002) Particle swarm optimization. In: IEEE international conference on neural networks, 1995. Proceedings, vol 1944, pp 1942–1948

Kopka H, Daly PW (2003) Guide to LATEX. Pearson Education, London

Larumbe F, Sanso B (2013) A tabu search algorithm for the location of data centers and software components in green cloud computing networks. IEEE Trans Cloud Comput 1:22–35

Li K, Tang X, Veeravalli B, Li K (2015) Scheduling precedence constrained stochastic tasks on heterogeneous cluster systems. IEEE Trans Comput 64:191–204

Liang A, Pang Y (2016) A novel, energy-aware task duplication-based scheduling algorithm of parallel tasks on clusters. Math Comput Appl 22:2

Livny J, Teonadi H, Livny M, Waldor MK (2008) High-throughput, kingdom-wide prediction and annotation of bacterial non-coding RNAs. PLoS ONE 3:e3197

Lo VM (1988) Heuristic algorithms for task assignment in distributed systems. IEEE Trans Comput 37:1384–1397

Lozovyy P, Thomas G, Simon D (2011) Biogeography-based optimization for robot controller tuning. In: Computational modeling and simulation of intellect: current state and future perspectives. IGI Global, pp 162–181

McCall J (2005) Genetic algorithms for modelling and optimisation. J Comput Appl Math 184:205–222

Mei J, Li K, Ouyang A, Li K (2015) A profit maximization scheme with guaranteed quality of service in cloud computing. IEEE Trans Comput 64:3064–3078

Rahmati SHA, Zandieh M (2012) A new biogeography-based optimization (BBO) algorithm for the flexible job shop scheduling problem. Int J Adv Manuf Technol 58:1115–1129

Ranaweera S, Agrawal DP (2000) A task duplication based scheduling algorithm for heterogeneous systems. In: Parallel and distributed processing symposium, 2000. IPDPS 2000. Proceedings. 14th International, 2000. IEEE, pp 445–450

Rarick R, Simon D, Villaseca FE, Vyakaranam B (2009) Biogeography-based optimization and the solution of the power flow problem. In: IEEE international conference on systems, man and cybernetics. SMC 2009. IEEE, pp 1003–1008

Shafei MAR, Ibrahim DK, El-Zahab EE-DA, Younes MAA (2014) Biogeography-based optimization technique for maximum power tracking of hydrokinetic turbines. In: 2014 international conference on renewable energy research and application (ICRERA). IEEE, pp 789–794

Shojafar M, Kardgar M, Hosseinabadi AAR, Shamshirband S, Abraham A (2016) TETS: a genetic-based scheduler in cloud computing to decrease energy and makespan. In: International conference on hybrid intelligent systems. Springer, pp 103–115

Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12:702–713

Topcuoglu H, Hariri S, Wu M (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 13:260–274

Wang L, Xu Y (2011) An effective hybrid biogeography-based optimization algorithm for parameter estimation of chaotic systems. Expert Syst Appl 38:15103–15109

Wang L, Arunkumaar S, Gu W (2002) Genetic algorithms for optimal channel assignment in mobile communications. In: Proceedings of the 9th international conference on neural information processing, 2002. ICONIP'02. IEEE, pp 1221–1225

Xie G, Li R, Li K (2015) Heterogeneity-driven end-to-end synchronized scheduling for precedence constrained tasks and messages on networked embedded systems. J Parallel Distrib Comput 83:1–12

Xu Y, Li K, He L, Truong TK (2013) A DAG scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization. J Parallel Distrib Comput 73:1306–1322

Xu Y, Li K, Hu J, Li K (2014) A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Inf Sci 270:255–287

Montage: an astronomical image engine (2006). http://montage.ipac.caltech.edu

Workflow gallery (2018). https://pegasus.isi.edu/workflow_gallery/

Workflow Generator (2006). https://confluence.pegasus.isi.edu/display/WorkflowGenerator

Yang T, Gerasoulis A (1994) DSC: scheduling parallel tasks on an unbounded number of processors. IEEE Trans Parallel Distrib Syst 5:951–967

Yogesh C, Hariharan M, Ngadiran R, Adom AH, Yaacob S, Polat K (2017) Hybrid BBO_PSO and higher order spectral features for emotion and stress recognition from natural speech. Appl Soft Comput 56:217–232

Zhang L, Li K, Li K (2015) Bi-objective optimization genetic algorithm of the energy consumption and reliability for workflow applications in heterogeneous computing systems. In: International conference on algorithms and architectures for parallel processing. Springer, pp 651–664

Zhou N, Qi D, Wang X, Zheng Z, Lin W (2017) A list scheduling algorithm for heterogeneous systems based on a critical node cost table and pessimistic cost table. Concurr Comput Pract Exp 29:e3944